SELF-ADAPTIVE DISSEMINATION OF CRITICAL DATA IN DYNAMIC WIRELESS NETWORKS

David Dorsey
Bjorn Jay Carandang
Moshe Kam
Drexel University
Data Fusion Laboratory
3141 Chestnut Street, Philadelphia, PA, U.S.A
{dave, jay, kam}@minerva.ece.drexel.edu

Chris Gaughan
Jose-Luis Sagripanti, Dr. Sc.
US Army RDECOM
Edgewood Chemical Biological Center
APG-Edgewood Area, Maryland 21010
chris.gaughan@us.army.mil

ABSTRACT

The distribution of data in large-scale dynamic wireless networks presents a difficult problem for network designers due to node mobility, unreliable links, and traffic congestion. In this work, we propose a framework for adaptive data dissemination protocols suitable for distributing data in large-scale dynamic networks without a central controlling entity. The framework consists of cooperating mobile agents and a reinforcement-learning component with value function approximation. A component for agent coordination is provided, as well as rules for agent replication, mutation, and annihilation. We examine the adaptability of this framework to a data dissemination problem in a simulation experiment and discuss potential benefits of this framework for the Future Combat System (FCS) Network.

1 INTRODUCTION

The large-scale, dynamic, and time-varying nature of operational environments for ad hoc wireless mobile and sensor networks present formidable challenges to the design of reliable dissemination protocols. Approaches based on centralized control over the network are often infeasible because they do not scale well and assume, unrealistically, a static structure in the form of routes or routing tree structures. Decentralized approaches that do not assume any structure in the network often rely upon a gradient that emerges as data flow toward a sink node. The problem with these structure-less approaches is that they assume the gradient is relatively static; all broadcasts originate from a small subset of nodes and all data flow to these same nodes. However, if the network is deployed for the purpose of communicating with mobile or static nodes working *within* the area of deployment, as is the case with networks envisioned for use by Brigade Combat Teams (BCTs) in the Future Combat System (FCS),

these traditional protocols are no longer suitable because any node may be the source of a broadcast to all other nodes.

As an example, consider a network that is deployed for use by first-responders battling a forest fire or in a military engagement where there is a threat of chemical or biological agents. The networks consist of mobile communication nodes and stationary sensors with wireless communication ability. In these scenarios, mobile networks are likely to be deployed with little or no existing infrastructure; the individual mobile and sensor nodes communicate through their wireless interface with other nearby nodes to form links. The effect of these local interactions among nodes is the emergence of a connected, *ad hoc* wireless sensor network over the area of deployment. The resulting network does not possess any global perspective; nodes are only aware of neighbors in communication range. The mobile nodes accompany the first-responders, warfighters, or unmanned ground vehicles (UGV) that move through the area of deployment. Mobile nodes may act as both sources and sinks of information. The goal is for the sensor network to provide the mobile nodes with updated information about the entire area of deployment while also acting as a medium for sharing information between mobile nodes. When a sensor is triggered or a mobile node sends a packet to a sensor node, the event is recorded and then broadcast to neighboring nodes. Ideally, the event is broadcast to the entire network so that when a mobile node comes in communication range of a sensor node, it may obtain updated event information from across the network.

An effective critical information dissemination protocol for use in such scenarios should possess, at minimum, the following characteristics:

1. Reduced energy usage and interference via suppression of redundant messages and minimization of protocol control data.

Report Documentation Page

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 2008	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Self-Adaptive Dissemination Of Critical Data In Dynamic Wireless Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Drexel University Data Fusion Laboratory 3141 Chestnut Street, Philadelphia, PA, U.S.A	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002187. Proceedings of the Army Science Conference (26th) Held in Orlando, Florida on 1-4 December 2008

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	**UU**	**8**	
unclassified	**unclassified**	**unclassified**			

2. Reliable data delivery and message prioritization (e.g., threat alarms receive higher priority than routine updates).
3. Adaptability to network dynamics (e.g., node failures, node mobility, congestion, and environmental RF noise).
4. Decentralized control i.e., no central controlling entity is used to control dissemination; the protocol is self-organizing and routing decisions are made locally.

Most of the current methods for disseminating data on wireless *ad hoc* sensor networks require a structure to be enforced on the network. By structure we mean a set of pre-established reliable links or routes; this includes hierarchical structures such as clusters, or flat structures such as trees. Gradient-based algorithms such as Gradient Broadcast (GRAB)(Ye, Zhong, Lu, and Zhang 2005) work well in situations where there is a static sink node to which all data are delivered. However, they are not designed to deal with situations where there is no pre-defined sink node or where the network topology is highly dynamic because they still require maintenance of the gradient.

Research in Swarm Intelligence(Bonabeau, Dorigo, and Theraulaz 1999) led to many algorithms for adaptive routing in dynamic networks using the approach from AntNet(Caro and Dorigo 1998). AntNet is an adaptive, distributed, protocol using mobile software agents to search for an optimal path between a source and a sink. The method is based on the ant colony metaphor, where "forward ants" are sent from the source node and, once they have located the sink node, they trigger the release of "reverse ants" that update the routing tables on the hosts between the source and the sink.

Other methods methods described in the literature that are designed to cope with dynamic networks employ a randomized approach; a wireless node, upon receiving a message, will rebroadcast to a random subset of its neighbors: these include work in gossip protocols(Hedetniemi et al. 1988), rumor routing(Braginsky and Estrin 2002), and epidemic routing(Eugster et al. 2004). These random approaches provide statistical bounds on reliability and energy usage derived using theory from random graphs and probability.

In the next section, we provide an overview of our proposed dissemination framework, followed by a more detailed description of the individual components of the framework. In Section 3, we provide experimental results from simulations. We conclude in Section 4 with a discussion of the potential benefits of the dissemination framework and future work. A more detailed presentation of the dissemination algorithm, theoretical background, and experimental results can be found in (Dorsey, Gaughan, et al. 2008).

2 PROPOSED METHOD

In this work, we propose an improvement over random strategies through the use of mobile software agents that learn and respond to changes in the network through the use of agent cooperation and reinforcement learning.

An agent is defined as anything that can perceive its environment through sensors and act upon the environment through actuators (Russell and Norvig 2003); mobile software agents are communication packets characterized by their ability to execute code on hosts. Our technique involves the use of mobile software agents that pick up and/or drop off data to nodes while recording information about the environment onto the nodes at each migration (i.e., a movement by an agent from one host to another). Agents make decisions about which data to pick up and which link to select for migration according to a state-action value function, which is formed by extracting and generalizing features from the local information recorded onto the nodes by other agents. They receive a reward for delivering an event to a host that has not already received the event. The reward is proportional to the number of events that are delivered at each migration and the value of the events (a time-decreasing function of the priority of the message). The agent is also penalized for consuming energy at each migration; the amount of energy that is consumed when an agent migrates is related to the number of events the agent is carrying. Thus, the objective for each agent is to quickly disseminate information without carrying redundant information.

Reinforcement learning techniques (e.g., Q-learning (Watkins and Dayan 1992)) typically characterize the interaction between the agent and the environment as a discrete time, finite state Markov decision process and update a set of value functions over all possible states or actions (stored in a large look-up table). Recent literature in multiagent reinforcement learning has considered the use of a function approximation to estimate the value function (Abul, Polat, and Alhajj 2000, da Motta Salles Barreto and Anderson 2008). In our propsed framework work, we use a linear approximation to the value function, which provides an enormous compression of the number of stored values an agent needs to keep while allowing the agent to make generalizations from states it has experienced about states it has not experienced. This approximation is of the form

$$\tilde{Q}(\ell,a) = w_0 + \mathbf{w}^T\boldsymbol{\theta}, \qquad (1)$$

where **w** is a vector of weights and θ is a vector of basis functions formed by extracting relevant features of the local environment. Agents use this value function to select a set of events to carry (a) and the next link (ℓ) to which the agent will migrate. The basis functions are formed using features including the amount of energy consumed for carrying a set of events, the delay experienced by other agents on a given link, and the link quality. These features are provided by messages left by other agents indicating the state of neighboring nodes and links. After each action, the agents update the weight vector in order to minimize the error between the expected value of a link and event set and the actual reward received. When an agents average reward increases above a threshold, it will replicate (clone) itself; agents whose average reward is below a minimum threshold are eliminated. Agents may also mutate weight vectors with agents who are performing better (weight vectors and average rewards are stored at the nodes in the agent messages).

2.1 Components of the Proposed Method

2.1.1 Events

We refer to any data that are produced by nodes in the network as an *event*. Events may originate from sensor detections or from user communications, such as an annotation on a collaborative whiteboard application used by first responders (for an example of such a collaborative application for first responders, see (Anderson, Dorsey, et al. 2004, Kopena et al. 2008)). Each event has a time-dependent numerical value determined by its age (how much time has passed since the event was detected), its priority class (higher priority events have greater initial values), and its expiration date (the amount of time for which an event is relevant). We denote the value of event i created at time t_0 with priority p and expiration x as $v_i(t - t_0, p, x)$ or simply $v_i(t)$. We denote the set of events on a host as E_h and the events that an agent is carrying as E_a.

2.1.2 Swarm Agents

There have been several investigations into the use of mobile software agents for exploring and providing decentralized services in wireless *ad hoc* networks (Massaguer et al. 2006), (Cicirello, Mroczkowski, and Regli 2005), (Peysakhov et al. 2004). A mobile agent is a composition of software and data which is able to autonomously move from one host to another while continuing its execution at each destination. Mobile agents are well suited to adaptive communication protocols because they are goal-oriented and can continue to operate even after the host from which they originate is removed from the network. The mobile agents used in the dissemination framework are composed of a link-action value estimator, $\tilde{Q}(\ell, a)$, a decision policy, π, a parameter set **w**, an event payload, and a fixed memory size for storing

important statistics. We refer to these as *swarm agents*. The link-action estimator computes the value of the agent choosing a link, ℓ, and an action, a. The choice of ℓ is selected from the set of neighbors of the current host as well as the current host (agents may elect not to migrate). The action is selected from the action space, A, which is comprised of the events the agent will carry to the next host, the number of clones to create, the decision to live or die, and the decision to mutate the agent's parameters, **w**, with another agent's parameters. Swarm agents collaborate through messages left on the hosts to evaluate links and actions.

2.1.3 Visit Entries

The messages left by swarm agents for collaboration are contained in a *visit entry* at each host they visit. These entries contain information about the agent's payload as well as information about the local environment as perceived by the agent. For example, the visit entries might contain:

- The quality of the link (*LQ*) upon which the agent migrated; this is a function of the received signal strength at this host and the number of migration failures the agent experienced on this link.
- The amount of energy the last host has consumed (*energy*).
- The amount of time the agent spent at this host and the last host it visited (*delay*)
- The events that the agent carried in its payload to this host and from this host (*events*)
- Event meta-data (*MD*) used to represent any data elements that the agent encountered at the last host but did not pick up
- The value of the events at the last host (*value*)
- The agent's parameter set, **w**
- The agent's average reward $\hat{R}$
- The number of times the agent has replicated (*clones*)
- The amount of time the agent has been alive (*age*)

The data in these visit entries are sorted into link advertisements and agent advertisements on the hosts, as shown in Tables 1 and 2. When an agent arrives at a new host after a migration, it will use the information in the link advertisements to determine the value of migrating across a link. If a link does not have an advertisement, then the agent will categorize the link as *unexplored*. Because agents always take the set of events that will maximize their expected reward, agents may elect to leave some events on hosts it has visited and take only the meta-data describing the event to the next host. The use of meta-data for exchanging information between nodes in a broadcast

protocol was first introduced in (Heinzelman et al. 1999).

The agent will also evaluate its own performance by viewing agent advertisements and comparing its own average reward with that of other agents. In (Haas, Peysakhov, and Mancoridis 2005), the authors use a genetic crossover method between swarms of randomly wandering agents on a network. The objective was to continuously mutate the agent populations in order to find an optimal memory size for agents that would result in the reduced latency of event delivery. In swarm dissemination, the agents mutate their parameter set $\mathbf{w}$; when an agent that has recently visited is performing significantly better (according to the advertisement), the agent may perform a crossover mutation between its own parameter set and that of the successful agent (Section 2.1.7). Visit entries and advertisements are deleted from the host after a set period of time to avoid crossovers and decisions based on stagnant information.

Table 1: Table of state values for each link to a neighbor of the current host. The values are updated periodically between hosts or through agent visit entries.

Link	LQ	events/ MD	value	energy	delay
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
...	-	-	-	-	-

Table 2: Table of state values for each agent to recently visit the current host. The values are updated through agent visit entries.

Agent	$\hat{R}$	$\mathbf{w}$	clones	age
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
...	-	-	-	-

2.1.4 Reward

At each time step, agents receive a reward R for delivering an event to a host that has not already received the event. The reward is proportional to the number of events that are delivered at each migration and the value of the events (which decreases with time). The agent is also penalized for consuming energy at each migration; the amount of energy that is consumed when an agent migrates is related to the number of events the agent is carrying. If an agent migrates with no events, it is still penalized a constant amount c to carry its executable code and its memory. If the agent is carrying N events, the reward is computed as

$$R = a_1 \sum_{i=1}^{N} I(v_i) - a_2 N + c \qquad (2)$$

where a_1, a_2, and c are constants, and $I(\cdot)$ is the indicator function

$$I(v_i) = \begin{cases} v_i & \text{if host did not have the event} \\ 0 & \text{otherwise} \end{cases}.$$

The agent keeps a running weighted average of these rewards $\hat{R}$ which is updated at each time step according to

$$\hat{R}(t) \leftarrow R(t) + (1 - \alpha)\hat{R}(t - 1) \qquad (3)$$

2.1.5 Link-Action Value Approximation

Swarm agents form the function approximation using the data contained in the link advertisements from Table 1. We refer to the data in the table as the *features*; each row contains features for a single link from the current host.

The first feature is the link quality, *LQ*, which represents the probability that an agent will be able to successfully migrate across this link. If an agent attempts to migrate, but the packet is dropped due to interference, then the agent will have to retransmit (and consume more energy at this host). The value of *LQ* is a function of the received signal strength, which indicates the strength of the signal received from the neighbor host, as well as the number of failed migration attempts reported by agents on this link. This function is computed so that more recent reports are given greater weight (this is true for all values in the advertisement table). The basis function θ_1 that computes this feature is a simple sigmoid function mapping the *LQ* value on the interval $[-1, 1]$.

The second feature is the *events/ MD*. This feature is the set of events that the neighbor on this link is known to possess as of the most recent visit entries; if the event is represented only by meta-data, then the event is possessed by the neighbor but not this host. With this information, two additional basis functions, θ_2 and θ_3 are formed. The first computes the maximum value of the events that the agent expects to "pick up" at this neighbor; the second is the maximum value of the events the agent expects to "drop off" upon migration. If hosts periodically update each other with the value of the events they currently possess, then the third feature, *value*, can be incorporated into θ_2 and θ_3 to verify the validity of the visit entry information. In the case where there is no information about this link from visit entries, an additional basis with a boolean value is used to indicate that this link is unexplored.

The *energy* feature represents the amount of energy known to have been consumed by the neighbor on a link. We used a sigmoid function to represent the value of this feature. The *delay* feature represents the amount of time agents spend at this neighbor. We assume in our experiments that if multiple agents reside on the same host that they are placed into a queue to wait until the CPU is available. The value also represents the congestion on this link. The basis function for this feature is the same as the one used for the *energy* feature.

The value of a link is computed with these basis functions using (), replacing s with ℓ. For each link, the action (set of events to carry to the neighbor on a link) that maximizes the value of $\hat{Q}(\ell, a)$ is denoted $\hat{Q}_\ell^*$.

After an agent has reached the next host, having selected action a and link ℓ and received reward R, the weights for the linear function are updated using:

$$w_i \leftarrow w_i + \alpha \left[r + \gamma \hat{Q}(\ell', a') - Q(\ell, a) \right] \theta_i(\ell, a), \quad (4)$$

Note that agents are also evaluating their link selection ℓ' and next action, a' with respect to the current state and the last action taken when updating $\mathbf{w}$. That is, agents update $\mathbf{w}$ *after* they have selected what link and action to take next and incorporate the estimated value of this decision into the learning update, as shown in Figure 1.

2.1.6 Migration Policy

The values for $\hat{Q}_{\ell_i}^*$ for each link ℓ_i are used to form a random distribution for the agent migration decision at each node. Two probabilities are calculated: the first is the probability of migrating this time step and the second is, given that the agent has chosen to migrate, the probability of selecting a particular link. In (Deneubourg, Goss, et al. 1989), the authors provide a model for examined patterns in army ant raids, where the ants leave the nest to seek out food sources. At each step, each ant decides whether to advance or stay at its current site. According to the model, if ρ_l and ρ_r are the amount of pheromone to the left and to the right of the ant in the forward direction, then the probability that the ant will advance is given by:

$$p_m = \frac{1}{2} \left[1 + \tanh \left(\frac{\rho_l + \rho_r}{100} - 1 \right) \right] \quad (5)$$

We adopt this model to determine the probability that an agent will advance in a given time step, where here the pheromone amounts are replaced with values for $\hat{Q}_{\ell_i}^*$. Let

N be the number of neighbors at node i. Then,:

$$p_m = \frac{1}{2} \left[1 + \tanh \left(\sum_{k=0}^{N} \hat{Q}_{\ell_k}^* - 1 \right) \right] \quad (6)$$

A model from the army ant raids (Bonabeau, Dorigo, and Theraulaz 1999) is also used to determine the probability of visiting a neighbor node. The probability of choosing link i is:

$$p_i = \frac{\left(\mu + \hat{Q}_{\ell_i}^* \right)^\beta}{\sum_{k=0}^{N} \left(\mu + \hat{Q}_{\ell_k}^* \right)^\beta} \quad (7)$$

where μ quantifies the degree of randomness involved in the decision and β determines the nonlinearity of the decision function. The values of p_i form a distribution P over which the next link is selected. A higher value of β implies that small differences in $\hat{Q}^*$ will result in large probability variations, so agents will be discouraged from choosing a link with even a slightly lower $\hat{Q}^*$. These parameters are used to adjust the agents' propensity for exploring new states or exploiting known information to maximize reward.

2.1.7 Evolution

Swarm agents are given a provision for reproduction, annihilation, and mutation. These operations are performed based on the value of the agents average reward $\hat{R}$ and the value of $\hat{R}$ reported by other agents in the agent advertisements (see Table 2). If an agent's reward is above a threshold r_{max}, then the agent will create a clone of itself (with identical parameters and events) with probability p. The number of possible clones increases linearly as $\hat{R}$ increases above r_{max} until reaching the number of links from this host. If the agent's reward is below r_{min}, then the agent will die with probability p. For reward values anywhere between r_{min} and r_{max}, the agent will compare $\hat{R}$ to the maximum average reward posted by an agent on the advertisement. If an agent advertised a reward that is higher than $\hat{R}$ at time t_a, then a mutation will occur with this agent with probability $p = e^{\alpha(t - t_a)}$. The mutation operation involves replacing random w_i from the agent's parameter set with the corresponding w_j from the advertised agent's parameter set. Specifically, if the successful advertised agent's reward is $\hat{R}_m$ and its parameter set is $\mathbf{v}$, each element $w_i \in \mathbf{w}$ is replaced with $v_i \in \mathbf{v}$ with probability $\frac{\hat{R}_m - \hat{R}}{\hat{R}_m}$.

The purpose of crossover is to allow agents who have already learnt the environment to share this information with other agents. When agents are receiving increasing rewards for dropping valuable data to hosts and minimizing

the number of unnecessary events in their payload, the agents will reproduce and share learnt behavior with other agents. This results in an increase in the agent population. When there are few opportunities for reward (either there are no data to be disseminated or most nodes already have the events), then agents will receive negative rewards and the agent population will diminish. The result is an agent population that grows and diminishes in proportion to the requirements of the network.

2.2 Algorithm

Figure 1 shows an overview of the swarm agent behavior cycle. Each time an agent migrates to a new host, it creates a visit entry and records the data described in Section 2.1.3. These data are added to the link and agent advertisement tables on the host. Then the agent merges its events with the events of the host and computes the received reward R and the average reward $\hat{R}$. Next, the agent checks if $\hat{R}$ is within $[r_{min}, r_{max}]$; if so, then it will look into the agent advertisement table and find $\hat{R}_{max}$ to evaluate whether or not to perform a crossover operation. If the reward is above the interval, the agent will perform a replicate operation. It will enter the die routine if the average reward is below the interval. Next, the agent will evaluate possible next links and actions using the link advertisement table and the value function $\hat{Q}(\ell, a)$ described in Section 2.1.5. This loop will return a value of $\hat{Q}^*_{\ell_i}$ for each link, which are used in the event/link decision, which is determined using the migration policy described in Section 2.1.6. Before migrating to the next host, the agent updates its parameters using (4).
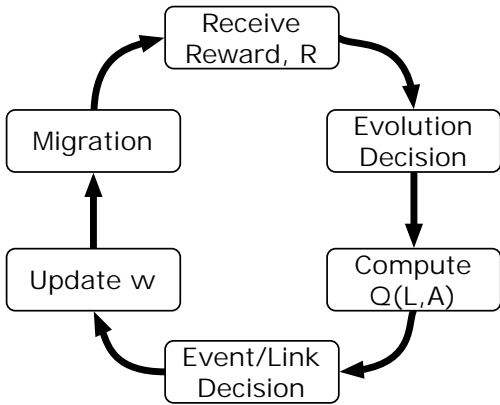

Figure 1: At each migration, an agent selects an action based on the current perceived state, receives a reward, and modifies the parameter set to minimize the difference between the perceived value of a state-action pair and the actual reward obtained for that action.
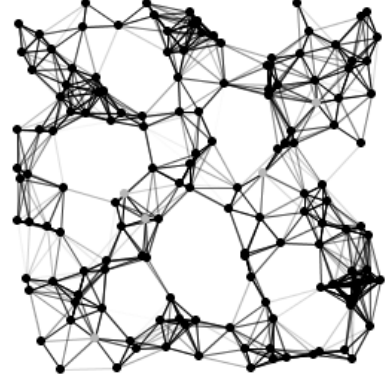
3   EXPERIMENTAL RESULTS

Figure 2: One sample simulation topology

We simulated the swarm agent dissemination algorithm using the Macro Agent Transport Event-based Simulator (MATES) (Sultanik, Peysakhov, and Regli 2006) on a 150 node network with a randomly generated topology for each experiment and 10 mobile nodes in each network. Figure 2 shows an example topology in the simulator. The Euclidean distance between two hosts was used to determine the link quality; the lines connecting nodes in the simulator denote neighbors (thinner lines represent lower link quality). Initially, 5 agents were randomly placed in the network and given a single event to be disseminated to all nodes. After running the simulator for 15 time steps, we added 5 more events onto the hosts. During each time step, we tracked the number of agents that were alive, the number of agents that were migrating, and the event coverage. Denoting the number of events on all hosts as S, the number of hosts as H, and the number of event types as N, the event coverage was calculated as $\frac{S}{HN}$. This experiment was run 10 times and the average was taken over the runs.

Figure 3 shows the average event coverage over time for the 10 runs. Notice that at 15 time steps, the coverage drops below 50%, since the number of event types doubled. The coverage then rapidly increases back to near 92% as new agents are produced to handle the dissemination requirements. Figure 4 shows how the number of agents in the network increases rapidly in response to low event coverage. Once the coverage percentage approaches the maximum, the number of agents quickly diminishes, since agents are unable to find opportunities for reward. At time step 15, when 5 additional events are introduced into the network, the agent population increases again until the event coverage reaches about 90%. Also note in this figure that at any given time step, many agents are not active. For example, near time step 7, the total agent population reaches 130, while only 85 are active. This is due to the fact that when the agent
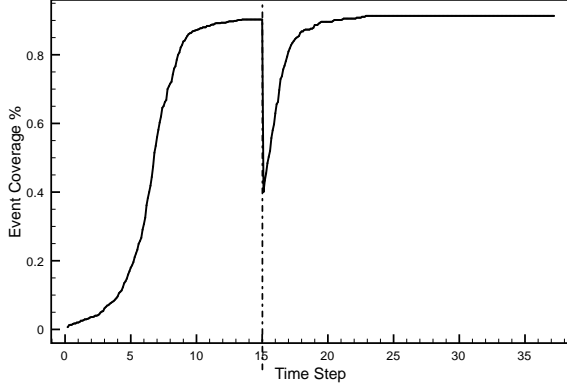
Figure 3: Event coverage over time. At 15 time steps, 5 additional events were added to the network, causing the coverage percentage to drop below 50%.

population grows quickly, some agents are either queued at a node waiting to gain access to the CPU or they are deciding not to migrate because there is little opportunity to gain reward in their location.
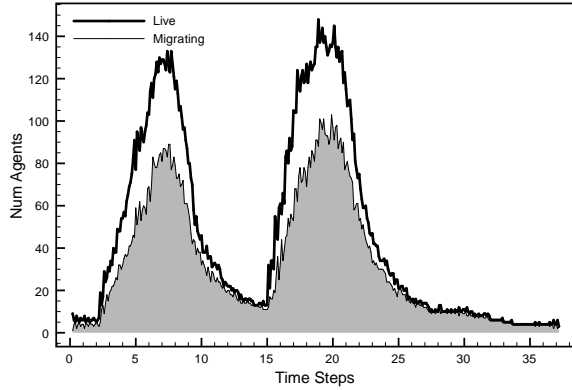

Figure 4: Number of agents on the network over time. The shaded area shows the number of agents that are migrating at any time; the bold line above shows the total number of agents in the network.

Figure 5 shows the average energy that was consumed at each host in order to cover a given fraction of the network with all of the events. These results were obtained using both static and mobile networks. For the static network, we ran 5 simulations with 5 events to be disseminated. The remaining two curves show energy versus coverage when all hosts are moving each iteration in a random direction. The "velocity" of the hosts was simulated by increasing the distance the host traveled in each iteration. We ran 5 simulations for each of two different velocities,

2 steps/iteration and 5 steps/iteration, where a step is 1/20 of the length of the simulation area. We note that higher mobility seems to reduce the amount of energy required to disseminate data; however, at 5 steps/iteration, the network became disconnected before full coverage could be obtained.
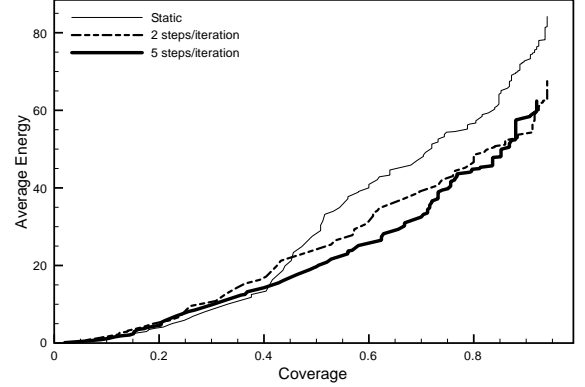

Figure 5: Average energy consumed at each host to cover a given fraction of the network with all of the events. The curves show averages over 5 simulations for a static network and networks with mobile nodes making random walks.

4   CONCLUSION

We have described a framework for self-adaptive dissemination of data in large dynamic networks using mobile software agents, inter-agent collaboration, and multiagent reinforcement learning. The learning component of the agent is a compact representation of the values of visited states which is used to provide generalizations about unvisited states. Agents share their experience by crossing their parameters with successful agents and sharing information about the environment in visit entries. We show that the agent population adapts to the needs of the network by reducing the number of agents in response to increasing event coverage and increasing the agent population to meet the dissemination requirements of the network. The agents who learn quickly and increase their reward reproduce and influence other agents, while agents who do not perform well are eliminated in order to reduce their impact on the network. The result is an autonomic dissemination support protocol that can be used to provide a decentralized dissemination service in *ad hoc* networks similar to those described in Section 1.

The focus of the FCS is a self-organizing, highly distributed communications network capable of gathering and disseminating information from and to warfighters, autonomous drones, and other devices in the network.

We believe the swarm dissemination framework provides a promising first step towards a reliable, secure, and decentralized data dissemination and data gathering service for FCS networks. The use of mobile agents eliminates the threat of vulnerable single points of failure in centralized communication hubs. The agents will be able to navigate around parts of the network that experience failure, and can continue to execute tasks after the host that created them has failed. In addition, since agents self-execute on hosts that provide agent support, new agent tasks can be added to the without reprogramming all nodes in the FCS network.

Future research in this direction will focus on improving the multi-agent reinforcement learning kernel. We have experimented with some of the parameters in the the the swarm algorithm, and have shown that the system is sensitive to various parameters. For example, increasing the probability for replication, p, will decrease the average delay experienced at hosts, while also increasing the energy consumption on the hosts. We will examine the set of tunable parameters in order to study performance trade-offs and patterns of stability, and focus on developing self-adaptive methods for tuning these sensitive parameters. The second focus of future research will be to develop bounds for stability and convergence on the global objective. We are specifically interested in showing whether agents may exploit a delayed global feedback signal in order to converge on optimal policies.

ACKNOWLEDGMENTS

REFERENCES

Abul, O., F. Polat, and R. Alhajj. Nov 2000. Multiagent reinforcement learning using function approximation. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 30 (4): 485–497.

Anderson, G., D. Dorsey et al. 8-9 April 2004. A secure wireless agent-based testbed. *Information Assurance Workshop, 2004. Proceedings. Second IEEE International*:19–32.

Bonabeau, E., M. Dorigo, and G. Theraulaz. 1999. *From natural to artificial swarm intelligence*. Oxford University Press.

Braginsky, D., and D. Estrin. 2002. Rumor routing algorthim for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, 22–31. New York, NY, USA: ACM.

Caro, G. D., and M. Dorigo. 1998. Mobile agents for adaptive routing. In *HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7*, 74. Washington, DC, USA: IEEE Computer Society.

Cicirello, V., A. Mroczkowski, and W. Regli. 30-31 Aug. 2005. Designing decentralized software for a wireless network environment: evaluating patterns of mobility for a mobile agent swarm. *Multi-Agent Security and Survivability, 2005 IEEE 2nd Symposium on*:49–57.

da Motta Salles Barreto, A., and C. Anderson. 2008, Jan. Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. *Artificial Intelligence*.

Deneubourg, J., S. Goss et al. 1989, Jan. The blind leading the blind: Modeling chemically mediated army ant raid patterns. *Journal of Insect Behavior*.

Dorsey, D., C. Gaughan et al. 2008. Self-adaptive dissemination of data in dynamic sensor networks. *Proceedings of the Second International Conference on Self-Adaptive and Self-Organizing System*:344–350.

Eugster, P. et al. 2004, Jan. Epidemic information dissemination in distributed systems. *Computer*.

Haas, J., M. Peysakhov, and S. Mancoridis. 2005. Ga-based parameter tuning for multi-agent systems. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, 1085–1086. New York, NY, USA: ACM.

Hedetniemi, S. et al. 1988, Jan. A survey of gossiping and broadcasting in communication networks. *Networks(New York*.

Heinzelman, W. R. et al. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 174–185. New York, NY, USA: ACM.

Kopena, J. et al. Jan.-Feb. 2008. Distributed coordination of first responders. *Internet Computing, IEEE* 12 (1): 45–47.

Massaguer, D. et al. 2006. Exploring sensor networks using mobile agents. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 323–325. New York, NY, USA: ACM.

Peysakhov, M. et al. 2004, Jan. Network awareness for mobile agents on ad hoc networks. *Autonomous Agents and Multiagent Systems*.

Russell, S. J., and P. Norvig. 2003. *Artificial intelligence: A modern approach*. Pearson Education.

Sultanik, E., M. Peysakhov, and W. Regli. 2006. Agent transport simulation for dynamic peer-to-peer networks. *Multi-Agent-Based Simulation VI*:162–173.

Watkins, C. J. C. H., and P. Dayan. 1992. Technical note: q-learning. *Mach. Learn.* 8 (3-4): 279–292.

Ye, F., G. Zhong, S. Lu, and L. Zhang. 2005, Jan. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*.